Object-Oriented php

You may have noticed that when defining (and using) php functions you need to explicitly declare the required inputs.

For example:

```
function subtract($x, $y) {
        $total = $x - $y;
        return $total;
}

print "The difference is ".subtract($test1, $test2);
```

Thus functions and their inputs are distinct entities that only come together when the functions are called.

This may be a risky strategy.

Object-Oriented php: Classes

The solution is to build a structure that combines the data with the functions that act on the data. In the object-oriented world this structure is called a class.

Each class is declared with a ‘unique’ name that indicates the purpose of the class. The keyword class is used to signify the declaration of a class.

```
class Arithmetic {
        …
}
```

Object-Oriented php: Classes

Next we declare the data of the class. These are declared as a set of named variables.

```
class Arithmetic {
        var $inp1;
        var $inp2;

        …
}
```

Object-Oriented php: Classes

We then declare the functions of the class. This should include a special function called __construct which is the constructor function.

This function is required when we want to use the class. It typically will assign values to the variables of the class.

```
class Arithmetic {
        var $inp1;
        var $inp2;

        function __construct($i1, $i2){
                $this->inp1;
                $this->inp2;
        }
}
```

Object-Oriented php: Classes

We then declare the functions of the class. This should include a special function called __construct which is the constructor function.

This function is required when we want to use the class. It typically will assign values to the variables of the class.

```
class Arithmetic {
        var $inp1;
        var $inp2;

        function __construct($i1, $i2){
                $this->inp1;
                $this->inp2;
        }
}
```

Object-Oriented php: Classes

We can now add the functionality of the class. That is, the functions that implement the behaviour of the class. In this case these are the arithmetic functions.

Object-Oriented php: Classes

We will do this by copying the (non-object-oriented definitions into the class and then editing…

```
class Arithmetic {
        var $inp1;
        var $inp2;

        function __construct($i1, $i2){
                $this->inp1;
                $this->inp2;
        }
        function add($inp1,$inp2) {
                $total = $inp1+$inp2;
                return $total;
        }
}
```

Note that the inputs to the function add already exist in the class.

Object-Oriented php: Classes

We will do this by copying the (non-object-oriented definitions into the class and then editing…

```
class Arithmetic {
        var $inp1;
        var $inp2;

        function __construct($i1, $i2){
                $this->inp1;
                $this->inp2;
        }
        function add() {
                $total = $inp1+$inp2;
                return $total;
        }
}
```

We may therefore remove the inputs to the function.

Object-Oriented php: Classes

We will do this by copying the (non-object-oriented definitions into the class and then editing…

```
class Arithmetic {
        var $inp1;
        var $inp2;

        function __construct($i1, $i2){
                $this->inp1=$i1;
                $this->inp2=$i2;
        }
        function add() {
                $total = $this->inp1+$this->inp2;
                return $total;
        }
}
```

However, $inp1 and $inp2 are now components of the class and hence must be accessed via $this.

Note how the function add acts on the data in its class.

Exercises…

Add the subtract, divide and multiply functions to the Arithmetic class.

Object-Oriented php: Objects

Classes are blueprints for objects. That is, a class specifies the data and functionality of one or more objects. An object is the instantiation of a class.

This is similar to the relationship between the architectural design of a house and the actual house. A class equates to the architectural design, and an object equates to the house. And of course there may be several houses that share the same architectural design.

Object-Oriented php: Objects

We may declare one or more objects of the class Arithmetic (or Arithmetic objects)…

This is done as follows…

```
$arith1 = new Arithmetic(25,50);
```

`$arith1` is a new variable that will host the Arithmetic object

`new` is a keyword that indicates that we are declaring a new object

`Arithmetic(25,50)` indicates that we are declaring a new Arithmetic object. This leads to the __construct function being called with the two inputs that have been provided.

The result is a new Arithmetic object whose data values are 25 and 50.

Exercises…

Declare three more Arithmetic objects: arith2, arith3 and arith4

Object-Oriented php: Objects

We may now use the objects.

```
$add1 = $arith1->add();

print "<p>The sum of the data in this object is ".$add1;
print "</p>";
```

Note that we have used the arrow notation again to access a component of an object.

```
$arith1->add()
```

Note also that the function add has no inputs since the data exists in the object $arith1.

Exercises…

Call the add function from each of the Arithmetic objects as in the following example:

```
$add1 = $arith1->add();
```

Use the print function to output the results of the applications of the Add function as in the following example:

```
print "The sum of the data in this object is ".$add1;
```

Exercises…

In the class Arithmetic define a printAdd function that assigns the result of the add function to a variable and then prints out the result.

Call the printAdd function from the four Arithmetic objects.